**Laboratory work №3**

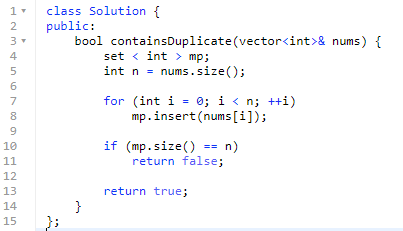
**Set and Maps**

1. **Contains Duplicate**

Given an array of integers, find if the array contains any duplicates.

Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

**Source code:**



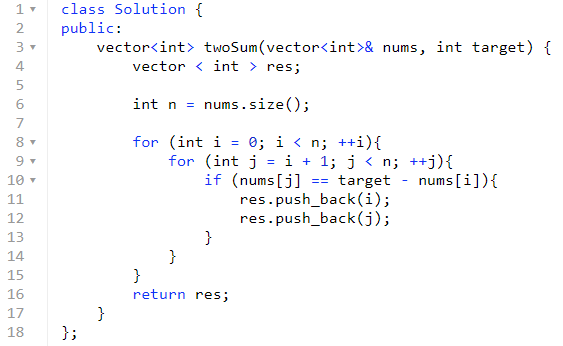
1. **Two Sum**

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly**one solution**, and you may not use the sameelement twice.

You can return the answer in any order.

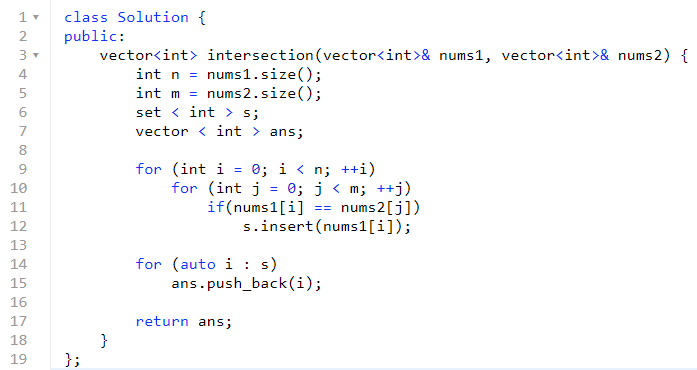
**Source code:**



1. **Intersection of Two Arrays**

Given two arrays, write a function to compute their intersection.

**Source code:**

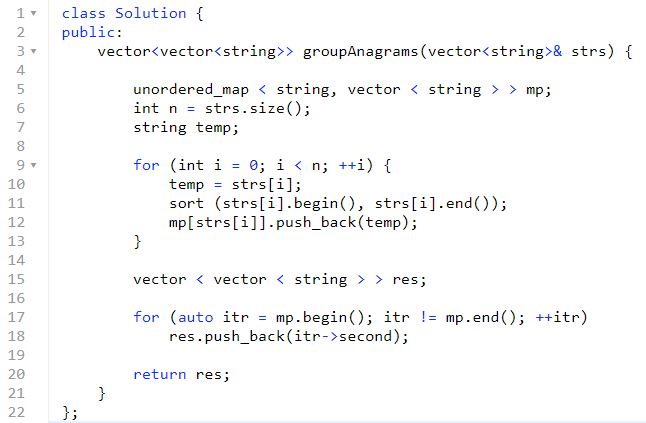


1. **Group Anagrams**

Given an array of strings strs, group **the anagrams** together. You can return the answer in **any order.**

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

**Source code:**

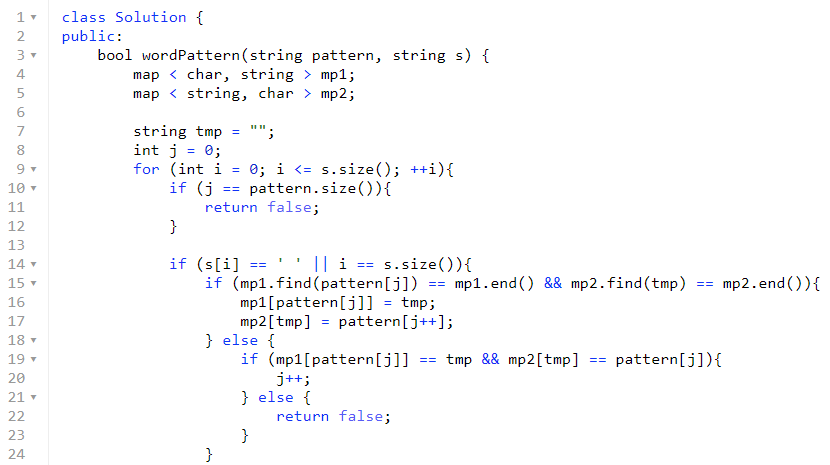


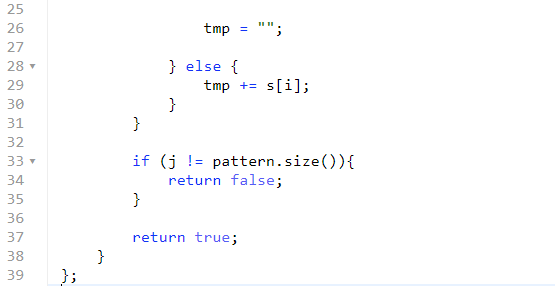
1. **Word Pattern**

Given a pattern and a string s, find if s follows the same pattern.

Here follow means a full match, such that there is a bijection between a letter in pattern and a non-empty word in s.

**Source code:**





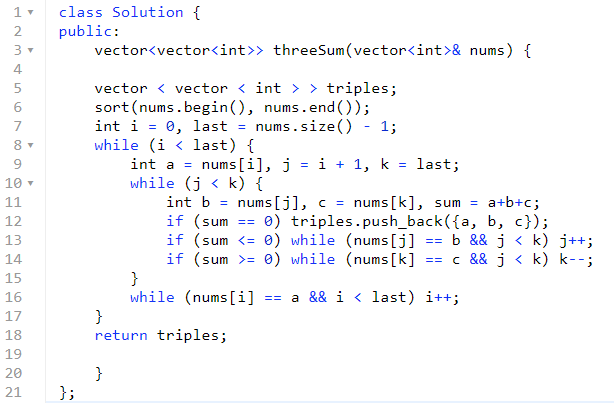
1. **Sum**

Given an array nums of n integers, are there elements a, b, c in nums such that a + b + c = 0?

Find all unique triplets in the array which gives the sum of zero.

Notice that the solution set must not contain duplicate triplets

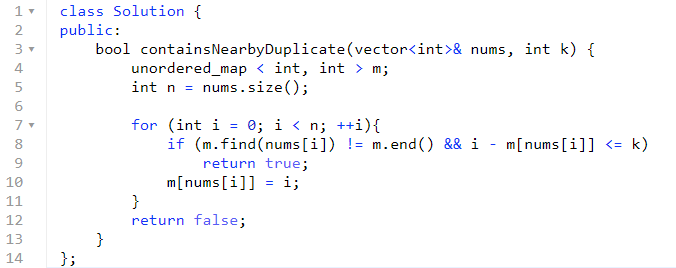
**Source code:**



1. **Contains Duplicate II**

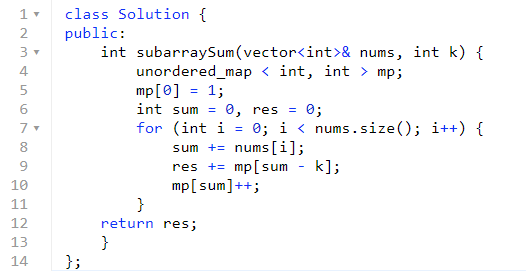
Given an array of integers and an integer *k*, find out whether there are two distinct indices *i* and *j* in the array such that nums[i] = nums[j] and the absolute difference between *i* and *j* is at most *k*.

**Source code:**



1. **Subarray Sum Equals K**

Given an array of integers and an integer **k**, you need to find the total number of continuous subarrays whose sum equals to **k**.



1. **Happy Number**

Write an algorithm to determine if a number n is "happy".

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1. Those numbers for which this process **ends in 1** are happy numbers.

Return True if n is a happy number, and False if not.